NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

INCORPORATION OF A DIFFERENTIAL GLOBAL POSITIONING SYSTEM (DGPS) IN THE CONTROL OF AN UNMANNED AERIAL VEHICLE (UAV) FOR PRECISE NAVIGATION IN THE LOCAL TANGENT PLANE (LTP)

by

Peyton M. Allen

March, 1997

Thesis Advisor: Isaac Kaminer

	Form Approved OMB No. 0704-0188
REPORT DOCUMENTATION PAGE	

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE INCORPORATION OF A DIFFERENTIAL GLOBAL POSITIONING SYSTEM (DGPS) IN THE CONTROL OF AN UNMANNED AERIAL VEHICLE (UAV) FOR PRECISE NAVIGATION IN THE LOCAL TANGENT PLANE (LTP)	5. FUNDING NUMBERS
6. AUTHOR(S) Allen, Peyton M.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)* The purpose of this thesis is to incorporate the Global Positioning System (GPS) and Inertial Navigation System (INS), for the guidance of an unmanned aerial vehicle (UAV) seeking precise navigation in a Local Tangent Plane (LTP). By applying the Differential Positioning technique, GPS position data becomes more accurate. This position can then be referenced to a known location on the ground in order to give the aircraft's position in the Local Tangent Plane.

The FOG-R UAV at the Naval Postgraduate School will be used for autonomous flight testing using a Texas Instruments TMS320C30 Digital Signal Processor (DSP). This DSP is hosted on an IBM compatible PC, and is controlled via Integrated System's AC100 control system design and implementation software package.

The GPS receiver used throughout this thesis is a Motorola PVT-6 OEM. Another identical GPS receiver is used as a reference station, thus providing the Differential capability. The objectives of this thesis are: the system must be able to accept current location from the GPS and convert it to LTP, display the LTP coordinates, numerically and graphically, and be able to easily change the origin coordinates. Finally, the achieved accuracy of the differential setup is examined.

14. SUBJECT TERMS *Differential Global Positioning System (DGPS), Unmanned Aerial Vehicle (UAV), Local Tangent Plane (LTP), Navigation	15. NUMBER OF PAGES 74
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18 298-102

INCORPORATION OF A DIFFERENTIAL GLOBAL POSITIONING SYSTEM (DGPS) IN THE CONTROL OF AN UNMANNED AERIAL VEHICLE (UAV) FOR PRECISE NAVIGATION IN THE LOCAL TANGENT PLANE (LTP)

Peyton M. Allen
Lieutenant, United States Navy
B.S., United States Naval Academy, 1989

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1997

ABSTRACT

The purpose of this thesis is to incorporate the Global Positioning System (GPS) and Inertial Navigation System (INS), for the guidance of an unmanned aerial vehicle (UAV) seeking precise navigation in a Local Tangent Plane (LTP). By applying the Differential Positioning technique, GPS position data becomes more accurate. This position can then be referenced to a known location on the ground in order to give the aircraft's position in the Local Tangent Plane.

The FOG-R UAV at the Naval Postgraduate School will be used for autonomous flight testing using a Texas Instruments TMS320C30 Digital Signal Processor (DSP). This DSP is hosted on an IBM compatible PC, and is controlled via Integrated System's AC100 control system design and implementation software package.

The GPS receiver used throughout this thesis is a Motorola PVT-6 OEM. Another identical GPS receiver is used as a reference station, thus providing the Differential capability. The objectives of this thesis are: the system must be able to accept current location from the GPS and convert it to LTP, display the LTP coordinates, numerically and graphically, and be able to easily change the origin coordinates. Finally, the achieved accuracy of the differential setup is examined.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to thank the people who contributed to this thesis. Dr. I. I. Kaminer for his guidance, teaching and patience. LCDR Eric Hallberg for his assistance in helping to solve the technical issues that arose. Jim Zanino for his computer knowledge. Dr. J. Clynch for providing the means and knowledge required to survey an accurate origin for our local tangent plane.

Most of all, I would like to thank my wife Debra for her constant support and understanding.

I. INTRODUCTION

Ever since the Global Positioning System (GPS) first became active, users have been attempting to make it more accurate and thus more useful. The civilian users only have access to a degraded signal with a nominal accuracy of one hundred meters, while the military has access to a signal that has an accuracy of about sixteen meters. By setting up a reference station at a known location, many of the errors in the civilian signal can be determined and thus a corrective signal can be sent out to all receivers within a geographical area. This is called differential GPS (DGPS) and its accuracy is easily within three meters. This level of accuracy makes it appealing to those who need precise navigation. However there are several problems even with DGPS: 1) Position information is updated only once a second, and thus can't keep up with large accelerations; 2) The signal is subject to temporary loss or degradation in accuracy for various reasons; and 3) The position still varies randomly within its "bubble" of accuracy. Due to all of these problems it is still necessary to integrate the DGPS inputs with the type of constant inputs found in an Inertial Measuring Unit (IMU).

The primary goal of this project was to provide the vehicle's current position (using the DGPS) in a local tangent plane in a manner such that it could be used to determine the necessary flight path required in order to follow a precise track over the ground. This information could then be integrated with INS data in order to send a continuous signal to the flight path controller. This system needed to be able to:

- Accept current location from the Differential Global Positioning System (DGPS) receiver and convert it to Local Tangent Plane (LTP) coordinates.
- Easily accept new origin coordinates for the Local Tangent Plane, to allow the system to be utilized in an arbitrary location.
- Display coordinates in the LTP in meters north, east, and down.
- Provide a graphical representation of the flight path relative to the origin of the LTP.

This could all be accomplished through the use of the MATRIX$_X$ Product Family of rapid prototyping software available from Integrated Systems Incorporated (ISI). This software utilizes a program called RealSim which uses a Graphical User Interface (GUI) to step the engineer through the design process. This thesis describes how these design requirements were met in the RealSim environment.

The Unmanned Aerial Vehicle (UAV) used for this thesis was the FOG-R UAV, commonly called the "Frog." It was acquired from the TEXCOM Experimentation Center at Fort Hunter Liggett, CA. It has a ten foot wingspan, a twenty pound payload capability, and is equipped with a full avionics suite, including Inertial Measuring Unit (IMU), Global Positioning System (GPS), and air data sensors. It is controlled through the use of a Radio Frequency (RF) link that sends a Pulse Width Modulated (PWM) signal which drives the aircraft's actuators. One of these links was modified and connected to the controller on the ground so the aircraft could be flown by the AC100 computer flight path controller system. A differential GPS was incorporated along with the requisite additional antennas to transmit and receive the correction signal to the GPS computed position.

II. THE GLOBAL POSITIONING SYSTEM OVERVIEW

In order to fully understand and evaluate the positioning information obtained by the GPS, its principles of operation will be discussed. The system consists of three major segments:

- The space segment
- The control segment
- The user segment

A. THE SPACE SEGMENT

The fully operational space segment consists of twenty-four satellites that are placed in six different orbital planes at an altitude of 10,898 nautical miles above the surface of the earth. Of these satellites, nine are Block II and fifteen are Block IIA. Three of them are on-orbit spares. They are all in twelve hour orbits arranged such that anywhere from five to eight satellites are visible from any point on the surface of the earth at any given time. [Ref. 1]

The information provided by the satellites to the users is sent to earth by means of two L-band carrier signals, L1 (1575.42 MHZ) and L2 (1227.6 MHZ). The L1 frequency carries the navigation message and the Standard Positioning System (SPS) code signals while the L2 frequency is used to measure the ionospheric delay by Precise Positioning System (PPS) equipped receivers. The navigation message is a 50 Hz signal consisting of data bits that describe the GPS satellite orbits, clock corrections, and other parameters. The Coarse Acquisition (C/A) code modulates the L1 carrier phase and is the basis for the civil SPS. The P-code (Precise) modulates both the L1 and the L2 carrier phases and is protected by encryption so that full positioning accuracy is denied to unauthorized users. Although all satellites are transmitting on the same frequency, they are each assigned their own unique C/A and P codes so that the GPS receivers can distinguish between them.

In an ideal world, the satellites should stay right on their orbits as they travel around the earth; unfortunately, this is not the case since their orbits are disturbed by various forces. The most important ones are: the gravitational perturbation of the earth, solar and lunar gravity, solar radiation pressure, and various gravity anomalies. Therefore, there exists the need for a control center that closely follows each satellite's behavior and takes the required corrective actions when necessary. This is the task of the control segment.

B. THE CONTROL SEGMENT

The control segment has the sole responsibility to make sure that the GPS satellites are in their proper orbit, functioning correctly, and transmitting the correct values of the navigational parameters. The GPS ground network consists of five active-tracking ground antennas and five passive-tracking monitor stations, located around the world.

The active-tracking ground antennas actively track the GPS satellites, transmitting commands and navigation uploads, and recording telemetry over S-band links. The passive-tracking monitor stations passively track the L-band signals transmitted by the satellites to determine the vehicle's navigational data.

All of the above data is collected at the Falcon Air Force Base, Colorado Springs, Colorado, where the master control station of the GPS control segment is located. The collected data is used as an input to a Kalman Filter from which the orbital states of the space vehicles can be determined. New navigational parameters are then passed to the satellites via the S-band radio links. The master control station personnel are also responsible for maneuvering the satellites to keep them in their preassigned orbits. When a satellite's performance is inaccurate or a mechanical failure has occurred, the vehicle is considered "unhealthy" and a special warning is incorporated in its navigational message to warn the GPS users.

With the whole GPS infrastructure perfectly working, the only remaining component required for a navigator to obtain a position fix is a GPS receiver which will receive, decode, and process the navigational data transmitted by the satellites. The user's end of the GPS, the GPS receiver, is referred to as the user segment.

C. THE USER SEGMENT

The user segment is responsible for obtaining the navigational signals provided by the satellites and extracting the precise values of three dimensional position, velocity, and time. There are several types of receivers. They are classified by how many channels they have.

Each separate channel is a radio receiver that can track one or more satellites. If it tracks one satellite, it is dedicated full time to that one satellite. If it tracks more than one satellite, it has to use a time sharing technique, that is, tracking each satellite sequentially for a specified amount of time. The time sharing technique yields poorer results and it was used by the very first receivers built. A typical number of separate channels for a modern receiver is six to twelve.

There are two major observables that can be used by a receiver in order to determine its position. The first is the pseudo-range and the second is the carrier phase. Velocity can be computed several ways also. The validity of these values can be affected by several things: accuracy of measurements, errors, and the addition of a differential station. The following is a discussion of each of these elements.

1. Pseudo-Range Navigation

Each satellite timetags its transmissions so that when they are received by the receiver, the latter knows the transmission time. The receiver also knows the time that the transmission was received from its internal clock. The receiver software then extracts the time difference between transmission and reception, that is, the time it takes the signal to travel from the satellite to the receiver. Multiplying this time difference by the speed of light will yield the actual distance between the satellite and the receiver.

Onboard the satellites, extremely precise and expensive atomic clocks keep track of GPS time. On the other end, the receivers carry relatively cheap clocks which are not very accurate. A one billionth of a second uncertainty in the time difference measurement multiplied by the speed of light yields a range uncertainty of one foot. So the receiver clock inaccuracy is an error that has to be taken care of during the range determination. The calculated range is called pseudo-range since it is not the actual range due to the receiver clock error.

By decoding the satellite message, the GPS receiver can also obtain the satellite's position (ephemeris) in earth-centered earth-fixed coordinates. By doing this for four satellites it can solve the following set of equations and determine its position coordinates (x, y, z) and the receiver clock error. All positioning coordinates (x,y,z) are given in earth-centered earth-fixed coordinate system:

$$R_1=\sqrt{(x_1-x)^2+(y_1-y)^2+(z_1-z)^2}+c*dt \tag{2.1a}$$

$$R_2=\sqrt{(x_2-x)^2+(y_2-y)^2+(z_2-z)^2}+c*dt \tag{2.1b}$$

$$R_3=\sqrt{(x_3-x)^2+(y_3-y)^2+(z_3-z)^2}+c*dt \tag{2.1c}$$

$$R_4=\sqrt{(x_4-x)^2+(y_4-y)^2+(z_4-z)^2}+c*dt \tag{2.1d}$$

where R_i is the pseudo-range, *dt* is the receiver clock error and *c* is the speed of light. The subscripts indicate the satellite being tracked.

Now the position fix of the receiver, which is obtained in earth-centered earth-fixed coordinates, must be transformed to the user's preferred coordinate system. It should be noted that the above equations represent a very basic problem formulation. In real life, the receiver's software has to take care of many other factors that are involved in the problem solution (such as satellite clock offsets and ionospheric delays). In fact many of the problem parameters are obtained as states of a Kalman filter, rather than as straight forward algebraic solutions.

2. Carrier Phase

Use of the carrier phase observable is a relatively new technique. When a GPS receiver acquires the signal of a satellite, it can measure the fractional part of a single cycle of the carrier wave. As every complete cycle represents distance equal to the wavelength of the carrier wave (20 cm for the L1 frequency), the carrier phase observable can be translated to range information from the particular satellite. Of course we cannot determine a pseudo-range only from carrier phase measurements because it is impossible to know how many complete cycles there are between the satellite and the receiver without any other information. What we can do is track the carrier phase changes, and feed them into the Kalman filter mentioned above to "smooth" pseudo-range estimates.

3. Accuracy

The accuracy of the Standard Positioning System is intentionally degraded by the Department of Defense (DOD) by modulating the position data with the Selective Availability (SA) code. According to the 1994 Federal Radionavigation Plan, the SPS signal has a predictable accuracy of 100 meters horizontal and 156 meters vertical. These are 95% accuracies, the value of two standard deviations of radial error. These values are not totally due to the SA coding on the signal; there are other errors added to the SA which add up to these figures.

4. GPS Error Sources

There are three categories of GPS errors: noise, bias, and blunders. The noise errors are the combined effect of PRN code noise (about 1 meter) and noise within the receivers (about 1 meter).

Bias errors have several causal factors. One is Selective Availability, which is the intentional degradation of the SPS signal by a time varying bias. SA was put into place by the DOD in order to limit the accuracy of GPS for non-U.S. military users. The C/A code is thus reduced from an accuracy of about 30 meters to a degraded accuracy of about 100 meters. Some of the other bias sources are: satellite clock errors uncorrected by the Control Segment (about 1 meter), ephemeris data errors (about 1 meter), unmodeled Tropospheric delays (about 1 meter), unmodeled ionospheric delays (about 10 meters), and multipath (about ½ meter). "Noise and bias errors combine, resulting in typical ranging errors of around fifteen meters for each satellite used in the position solution."[Ref. 1]

Blunders can result in large errors on the magnitude of hundreds of kilometers and can come from a variety of sources. There are human errors such as Control Segment mistakes or the user choosing the wrong geodetic datum. Then there are receiver errors which can be either hardware or software related. Essentially, a blunder is a failure of some part of the total system to operate as designed.

5. Geometric Dilution of Precision (GDOP)

GDOP is not defined as an error but rather as a "factor" in the accuracy of the GPS data. GDOP is a measurement of the respective angles between the satellites and the user. It is defined as being inversely proportional to the volume of the shape described by the unit-vectors from the receiver to the satellites used in a position fix. Thus a large unit-vector-volume will produce a low value of GDOP which is good, while a poor GDOP will result when the angles from the receiver to the set of satellites used are small.

III. SETUP OVERVIEW

For an in-depth discussion of each piece of the hardware and software utilized in this project see Chapter 3 of Ref. 2.

There are two major sets of components associated with this thesis:

- The airborne components
- The ground components

Each of these will be looked at for their contribution.

A. AIRBORNE COMPONENTS

The largest of these is quite obviously the Unmanned Aerial Vehicle (UAV) itself. The Frog is a high-wing, high-engine, tricycle gear aircraft with both forward and mid-fuselage payload sections. It is equipped for autonomous flight with an avionics suite. This suite consists of the following components: an IMU, air data sensors, and a Motorola PVT-6 DGPS receiver. These devices transmit their data to the ground station and receive the differential corrections via two spread spectrum RF links. In addition there is a Pulse Code Modulation (PCM) receiver that is used to drive the actuators on the control surfaces and the throttle. The Motorola PVT-6 receiver was mounted in the mid-fuselage section and connected to the antenna which was mounted on the top side of the empennage midway to the tail.

B. GROUND COMPONENTS

The majority of the equipment was kept on the ground for several reasons: space and weight restrictions on the aircraft, ease of operating and maintaining, and of course necessity. The ground station handles all of the flight management functions and data collection. It consists of the following three components:

- Sparc II Workstation: this computer contains the software package RealSim that is used to design, code, and implement a control algorithm.
- Luggable PC: this computer contains the TI C30 digital processor, and the DSP_FLEX board that holds the IP_Modules. The IP_Modules are discussed in Ref. 2, Chapter 3, Part B.
- Communications Box: this box houses various devices used to communicate with the UAV.

The differential antenna had to be on the ground at a known location in order to provide an accurate reference point. Here it was connected to the second of the Motorola PVT-6 receivers which was housed in the communications box. The other items in this box were the following: two DGR-115 spread spectrum RF modems which were used to transmit and receive telemetry information, and a Futaba PCM transmitter which had been modified to give the computer the ability to control the aircraft. This communications box was then hooked up to the luggable PC via four ribbon cables from the four IP_Modules. This computer actually controlled the aircraft and received the aircraft data output. It was this data which was decoded and sent to the Sun workstation via a standard TCP/IP connection. Here it could be easily utilized by the RealSim software for viewing of the UAV's position and flight parameters.

IV. COORDINATE TRANSFORMATIONS

In order to utilize the data from the GPS receiver, the data first needs to be converted to a more useful format. The data for latitude and longitude is received as milliseconds of degrees and it is most useful as meters north, east and down as defined by the LTP.

Since the receiver puts out latitude and longitude in milliseconds of degrees, the first step is to convert this into degrees of latitude and longitude, the Geodetic Coordinate System. In order to do this, the milliseconds are divided by 3,600,000, the number of milliseconds per degree.

The second step is to use a coordinate transformation to convert to a cartesian coordinate system called the Earth-Centered Earth-Fixed (ECEF) frame. (For a review of coordinate transformation matrices see Ref. 3.) In order to make this conversion, there are several other values which need to be known: h (altitude in meters) and N (the length of the ellipsoidal normal from the ellipsoidal surface to its intersection with the ECEF z-axis in meters). As shown in Chapter 2 of Ref. 3, N can be determined from the eccentricity factor (ϵ), the local latitude(ϕ), and the semi-major axis of the Earth ellipsoid (a).

$$N = \frac{a}{\sqrt{(1 - \epsilon^2 \times \sin^2(\phi))}} \tag{4.1}$$

Using this information, and introducing longitude (λ), one can then obtain the transformations from a geodetic to a cartesian representation of a position in the ECEF coordinate system as:

$$x=(N+h)\cos(\phi)\cos(\lambda) \tag{4.2a}$$

$$y=(N+h)\cos(\phi)\sin(\lambda) \tag{4.2b}$$

$$z=[N(1-\epsilon^2)+h]\sin(\phi) \tag{4.2c}$$

Now the transformation from ECEF to LTP must be applied; however, the LTP cannot be defined without an origin so this point must also be resolved in ECEF. In order to determine location in the LTP, a vector can be built from this origin to the point by subtracting the point's x, y, and z coordinates from those of the origin. It is this difference vector which will then be converted to give LTP coordinates, by multiplying it with a rotation matrix — R, from ECEF to Tangent plane. This transformation will involve one rotation about the z-axis and one rotation about the y'-axis, which will result in an Up, East, North axis orientation. This will then be converted to a North, East, Down system with one more matrix multiplication.

First the rotation about the z-axis is through the angle λ and results in this matrix:

$$\begin{bmatrix} \cos(\lambda) & \sin(\lambda) & 0 \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

The next rotation is about the new y-axis and is through the angle ϕ, resulting in:

$$\begin{bmatrix} \cos(\phi) & 0 & \sin(\phi) \\ 0 & 1 & 0 \\ -\sin(\phi) & 0 & \cos(\phi) \end{bmatrix} \tag{4.4}$$

When these two are multiplied together they result in an Up, East, North {UEN} coordinate system defined by:

$$\begin{bmatrix} \cos(\phi)\cos(\lambda) & \cos(\phi)\sin(\lambda) & \sin(\phi) \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ -\sin(\phi)\cos(\lambda) & -\sin(\phi)\sin(\lambda) & \cos(\phi) \end{bmatrix} \tag{4.5}$$

Since we desire a North, East, Down {NED} system (commonly used for aircraft applications), the above matrix must be multiplied by a matrix which can swap the first and last row and make the new last row the negative of the old first row. This is the required matrix

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{4.6}$$

When the previous two matrices are multiplied together one comes up with the required transformation matrix R, which needs to multiply the position difference vector in order to complete the transformation from the ECEF coordinate system to the LTP system (T).

$$(x\ ,\ y\ ,\ z)_T = {}_E^T R \times (\Delta x\ ,\ \Delta y\ ,\ \Delta z)_E \tag{4.7}$$

$$Here\ :{}_{E}^{T}R = \begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\phi)\sin(\lambda) & \cos(\phi) \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ -\cos(\phi)\cos(\lambda) & -\cos(\phi)\sin(\lambda) & -\sin(\phi) \end{bmatrix} \tag{4.8}$$

ϕ = *latitude of the origin*, λ = *longitude of the origin*

This will result in a three-dimensional vector describing the location of the aircraft in the Local Tangent Plane in meters from the origin. This particular rotation matrix results in a North, East, Down {NED} reference frame from the origin.

These computations were implemented in the MATRIX$_X$ software package. First the code was written in XMATH in order to validate the equations, and then transferred to the SystemBuild environment. Here it was rewritten as a series of Superblocks so that it could be easily inserted into the already designed SystemBuild controller for the aircraft. In RealSim a Graphical User Interface (GUI) page was created in order to allow for easy editing of the origin point so that this controller can be used anywhere there is a known position on which to set the differential antenna. This GUI also allows the current location of the aircraft in the LTP to be viewed both numerically and graphically on two 2-dimensional plots (x vs y, and y vs z). (See App. A)

V. CALIBRATION AND VALIDATION

Calibration was carried out in order to determine the accuracy of this DGPS configuration and the validity of the equations and data. This was done prior to implementing the code into the controller in order to ensure that troubleshooting could be done in an expeditious manner. This chapter will deal with first the calibration and then the validation.

A. CALIBRATION

The calibration was conducted in the parking lot of the Mechanical Engineering (ME) building on campus. In this parking lot are several markers for surveyed points placed by Dr. Clynch of the Oceanography Department. (See App. B) The differential receiver's antenna was placed over marker NPS-5005 for all of the data sets.

There were four sets of data collected from a static position and two sets of moving data collected. The data collected from the stationary runs is summed up in Table 5.1. The first two (#1 and #2) of the four stationary sets were taken with the aircraft co-located with the differential antenna. Each of these had a valid differential, three dimensional fix the entire time. The remaining two stationary sets (#3 and #4) were taken with the aircraft at position NPS-5003 and with the differential antenna remaining at position NPS-5005. These also had a valid differential, three dimensional fix.

The data set collected with the antennas virtually on top of each other had a constant value with variations so small as to be insignificant (Figures 5.1 and 5.2). The two stationary sets taken at a distance (Figures 5.3 and 5.4) did have some deviation in their values over time; however, the horizontal accuracy was still under 2 meters. The mean and standard deviation values here can be looked at in a variety of axes since the values start as Geodetic coordinates, are transformed to ECEF coordinates, and then converted to LTP coordinates. The errors that we are most interested in are those in the LTP, since this will be the coordinate system in which we are will be programming and monitoring the vehicle's flight. As the numbers show, the system is fairly accurate in the horizontal plane of this navigation frame and is slightly less accurate in the vertical plane of this navigation frame. This is to be expected since the errors are almost twice as bad vertically as opposed to horizontally for GPS. [Ref. 4]

	Run #1	Run #2	Run #3	Run #4
North sigma	1.465e-14	1.910e-14	1.527	1.477
East sigma	1.776e-15	8.660e-15	1.477	1.558
Down sigma	4.263e-14	1.776e-15	4.037	2.815

Table 5.1 Values of deviation in LTP axes for stationary runs (meters).

The values for the moving data sets are listed in Table 5.2. The standard deviation here was around one meter with a mean error from the path of less than 1.5 meters. The sample size for these measurements was small compared to the stationary (less than 100 data points versus around 800 data points); however, the data does show that the system will maintain its accuracy while in motion and not necessarily degrade to an unacceptable status. Runs #5a and #6a (Figures 5.5 and 5.7) were conducted from point NPS-5005 to point NPS-5004, while runs #5b and #6b (Figures 5.6 and 5.8) were the return trip. To obtain this data, the UAV was pushed by hand between these two points. However, there was no line marked out on the ground and thus some additional errors could have been induced. It can be noted that both of the run-a columns have a greater mean error than the run-b columns; part of this error was likely self-induced. The marker at point NPS-5004 was not as visible as the differential antenna parked on top of point NPS-5005, and thus was more difficult to walk a straight line towards. Lesson learned - paint the lines on the ground so as not to induce extra errors.

	Run #5a	Run #5b	Run #6a	Run #6b
North sigma	1.301	0.720	1.065	1.596
East sigma	0.876	0.520	1.157	0.820
Down sigma	1.957	0.929	2.935	0.523
Mean error	2.529	1.271	2.816	0.738

Table 5.2 Values of deviation in LTP axes for non-stationary runs (meters).

B. VALIDATION

In the validation phase, a program was written to compute the coordinate transformation from geodetic coordinates to ECEF coordinates. Prof. Clynch's data for the reference points in the Mechanical Engineering parking lot was run through this program "clynchck.ms" (see App. C), and compared to the values which he listed for the x, y, and z coordinates (see App. B). Once this conversion was validated, two points were then used to validate the code for the conversion from ECEF to LTP. One point was input as the origin and the other as an aircraft position, in order to check the LTP transformation and ensure that the proper distances and directions were being returned. All of the conversions matched with Dr. Clynch's down to the millimeter, at which point his numbers were rounded off. The LTP coordinates showed correlation to ± 1 millimeter horizontally and vertically. This served to prove the accuracy of the program for both coordinate transformation and LTP calculation. These results can be seen in App. D.

VI. FLIGHT TEST

A. FLIGHT TEST SETUP

In order to utilize the LTP data supplied by this project during a flight, there are two initial conditions which must be configured prior to takeoff. The coordinates for the origin of the LTP must be confirmed in both the RealSim software and in the coding for the DGPS receiver. Once all of the rest of the pre-flight setup has been completed then the following procedure can be used to ensure that the Local Tangent Plane position of the UAV will be computed correctly. All of the computations assume that the Differential antenna is the location of the origin of the LTP.

In order to check the coordinates in the RealSim software one follows this procedure. From the GUI Master page select the GPS_LTP page and view the coordinates displayed for the latitude, longitude, and geoid height of the origin. If these coordinates are correct then the next step is to check the coordinates of the DGPS receiver; if they are not correct, then they need to be updated. This is accomplished by double clicking in the upper right hand corner of each of the three coordinate display boxes and thus opening up the data input menu for each of these. One of the selections will be to change the initial value of the box; this is where the old value will be shown. Delete this value and type in the new value. Select the “Done” option once the value is correct and then proceed to the next box. When all three values are correct, check the DGPS receiver.

To change the values for the location of the DGPS receiver, which needs to be done in order to receive proper corrections for the LTP values, the luggable PC will be used. At the main C prompt, type in “cd GPS40", then from that directory type “GPS.” At the next prompt the user will be asked to enter a commport to be used; select commport 2. Now for the position to be displayed enter “pos:1". The next few steps are for actually entering the location information. The following command is found in Ref. 5 on page 36. From the GPS40 directory main page, type in the **php** (position-hold position) command, which will be the following:” **php:**xxx_xx_xx.xxx_yyyy_yy_yy.yyy_zzz.zzz_www” Where latitude is represented by the x, longitude by the y, height by the z, the w is for “GPS” or “MSL” depending on which type of height is being used, and the “_” represents a space. Latitude and Longitude both need to be entered as “deg_min_sec.thousandths of seconds” with the very first position being a “+” or “-” to indicate proper hemisphere. Height is entered in “meters.thousandths of meters.” This format is not the same as the manual suggests. However, it was determined through trial and error that this is the format which needs to be used.

B. FLIGHTS AT CHUALAR

For flights that take place at Chualar remote control (RC) airfield, which is where all flights so far have occurred, the LTP origin location is already entered in the appropriate places. Now all that remains is to ensure that the differential antenna is placed on top of the appropriate spot.

There is a six inch aluminum disk, called a survey marker, cemented in to place on the ground in a location which has been surveyed to an accuracy of ±9 cm. This survey was accomplished with the help of Dr. Clynch and his GPS equipment. The disk is found by standing at the northwest corner of the awning, facing the runway and then measuring off about 24 feet on a heading of 035° and looking down. The disk has been stamped 'CHUALAR-1'. This is the spot over which the differential antenna should be placed. In the event that flights are conducted elsewhere and then resumed at Chualar, the coordinates for the location of the disk are given below:

Latitude:	36.55020693 degrees
Longitude:	-121.54329221 degrees
Height, Geoid:	-5.67 meters

C. FLIGHT TEST DATA

During the flight testing conducted on 31 January 1997 at Chualar RC field, the pilot, Don Meeks, was asked to perform a circle in order to store this data for later analysis. This data was run through the LTP conversion program ('e2tang.ms' found in App. E) and was plotted (Figure 6.1 of App. A). As can be seen, he maintained a fairly constant circle of approximately 230-250 meters in diameter and deviated in altitude by only 34 meters over a period of almost 45 seconds. The graph of north vs. east (in Figure 6.1) is one of the two 2-dimensional graphs which will be used for real-time display and analysis for future flights. It can be used to monitor flight testing parameters and thus enable more meaningful data collection and interpretation.

VII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

This thesis has met the primary goal of providing the UAV's current position (from DGPS) in a local tangent plane in a manner such that it could be used to determine the necessary flight path required to follow a precise track over the ground.

This was done using the MATRIX$_X$ rapid prototyping software. The Interactive Animator (IA) was used to design a GUI in which the origin coordinates could be easily entered and/or changed. The IA page that was created also displays the UAV's real-time LTP position on two 2-dimensional graphs.

These LTP coordinates are supplied by the RealSim coded transformation of the DGPS provided geodetic coordinates. This transformation has been validated by making multiple conversions on surveyed data points and comparing the output to the survey values.

Finally, this portion of the controller has been tested in flight and has demonstrated the conversion to LTP coordinates and the real-time graphical representation of the flight-path in the local tangent plane.

B. RECOMMENDATIONS

The next step would be to find a place with known coordinates for a line, like either the ME parking lot or the runway of Chualar field, and conducting multiple runs at various speeds and checking the accuracy of the data. This would help to determine if the velocities at which the Frog is being flown will allow it the required accuracy to be flown by the *AC100* system all the way to landing. A separate project would be the use of this transformation in complex trajectory programming and tracking in the LTP.

APPENDIX A. FIGURES

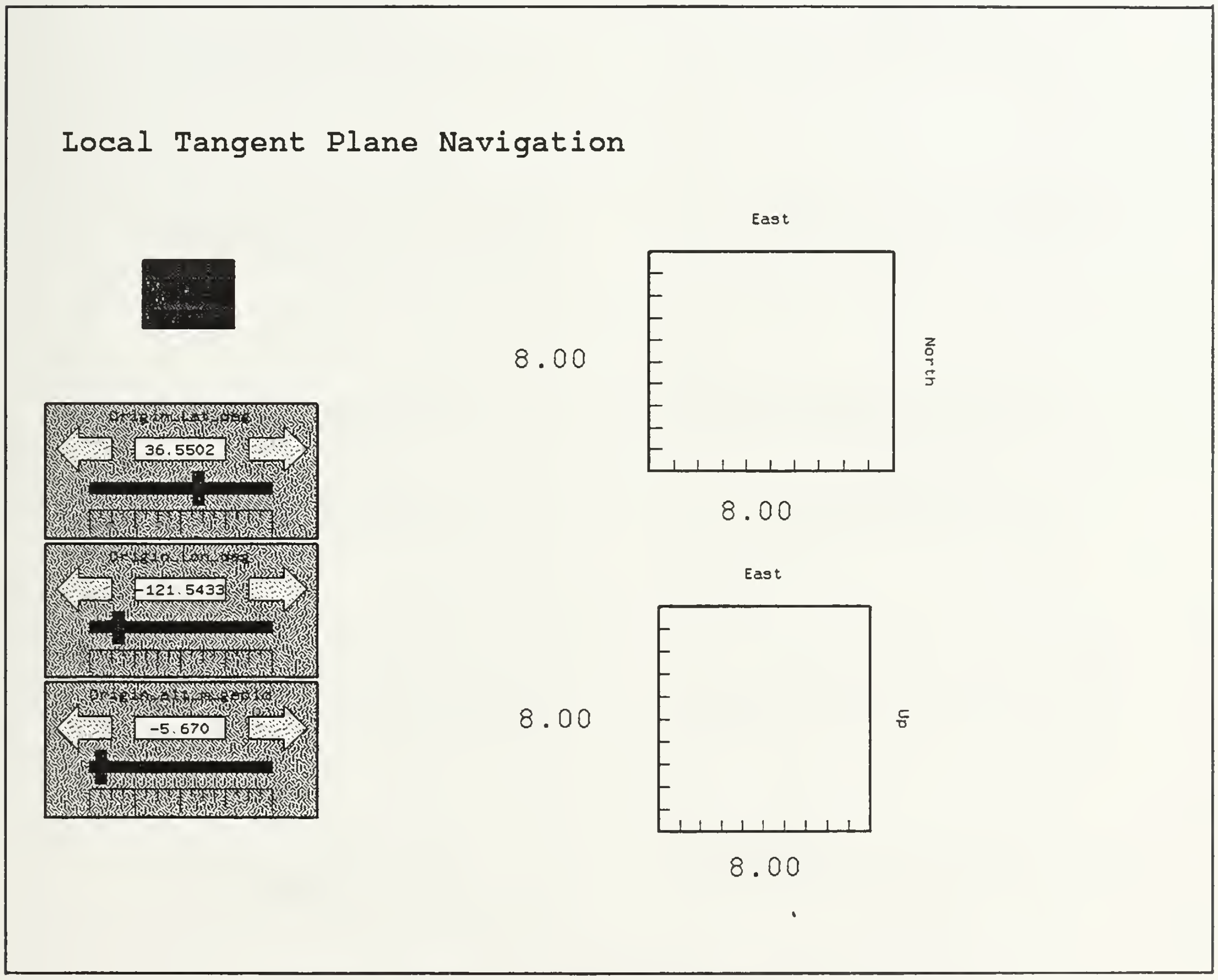

Figure 4.1: Interactive Animation display page

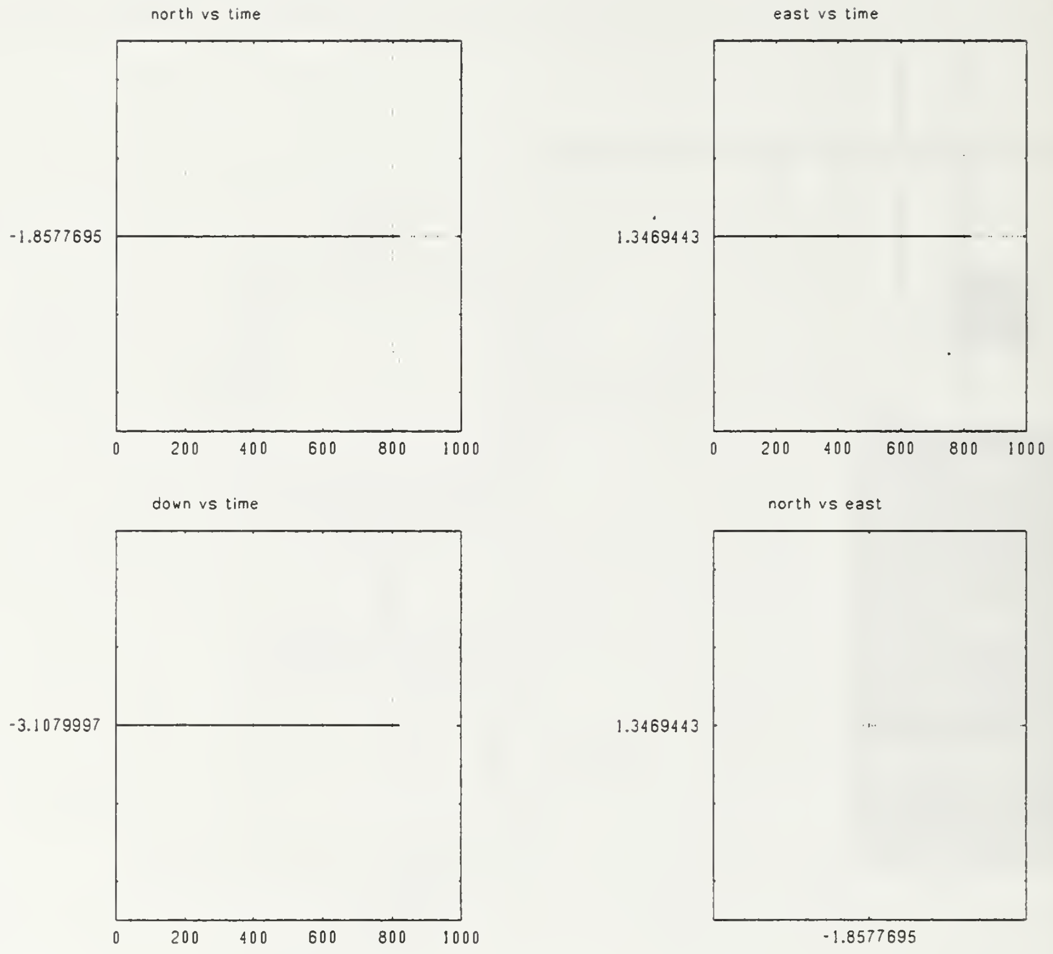

Figure 5.1: LTP data for run #1, all vertical axes in meters

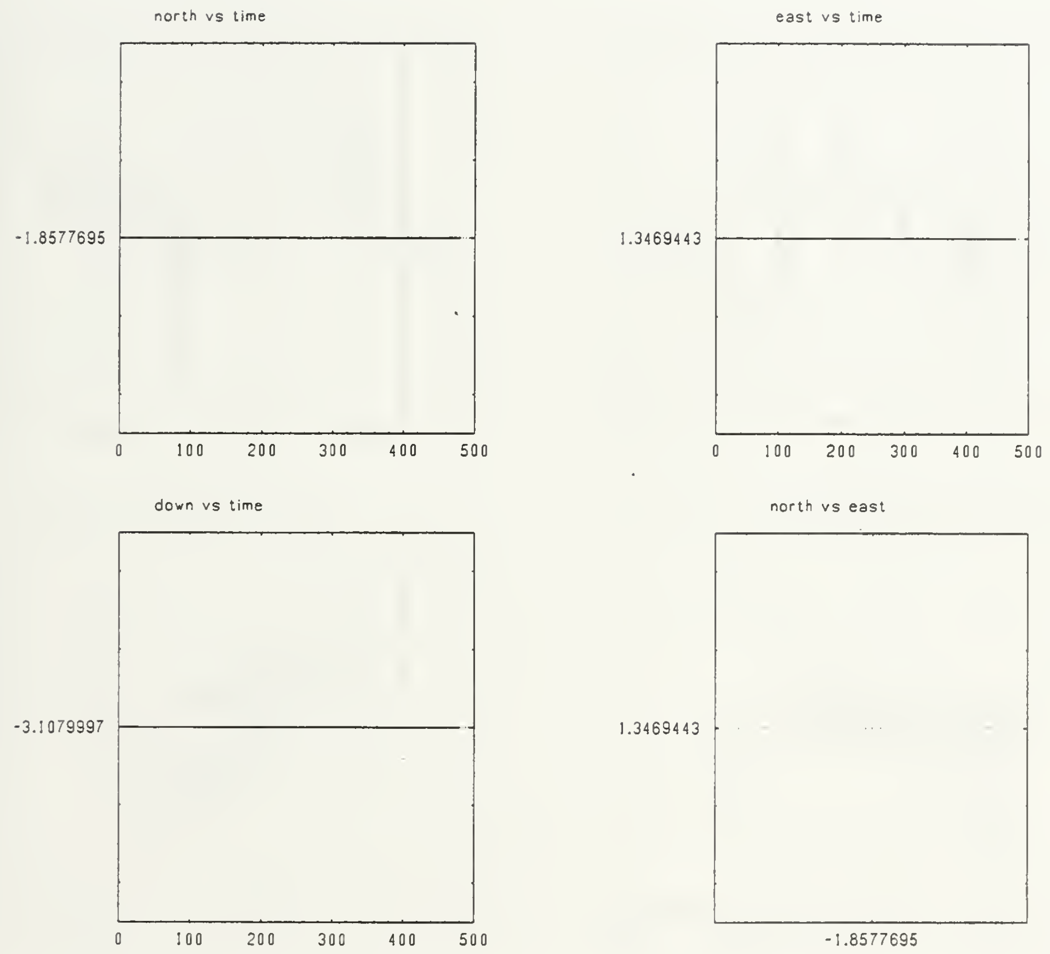

Figure 5.2: LTP data for run #2, all vertical axes in meters

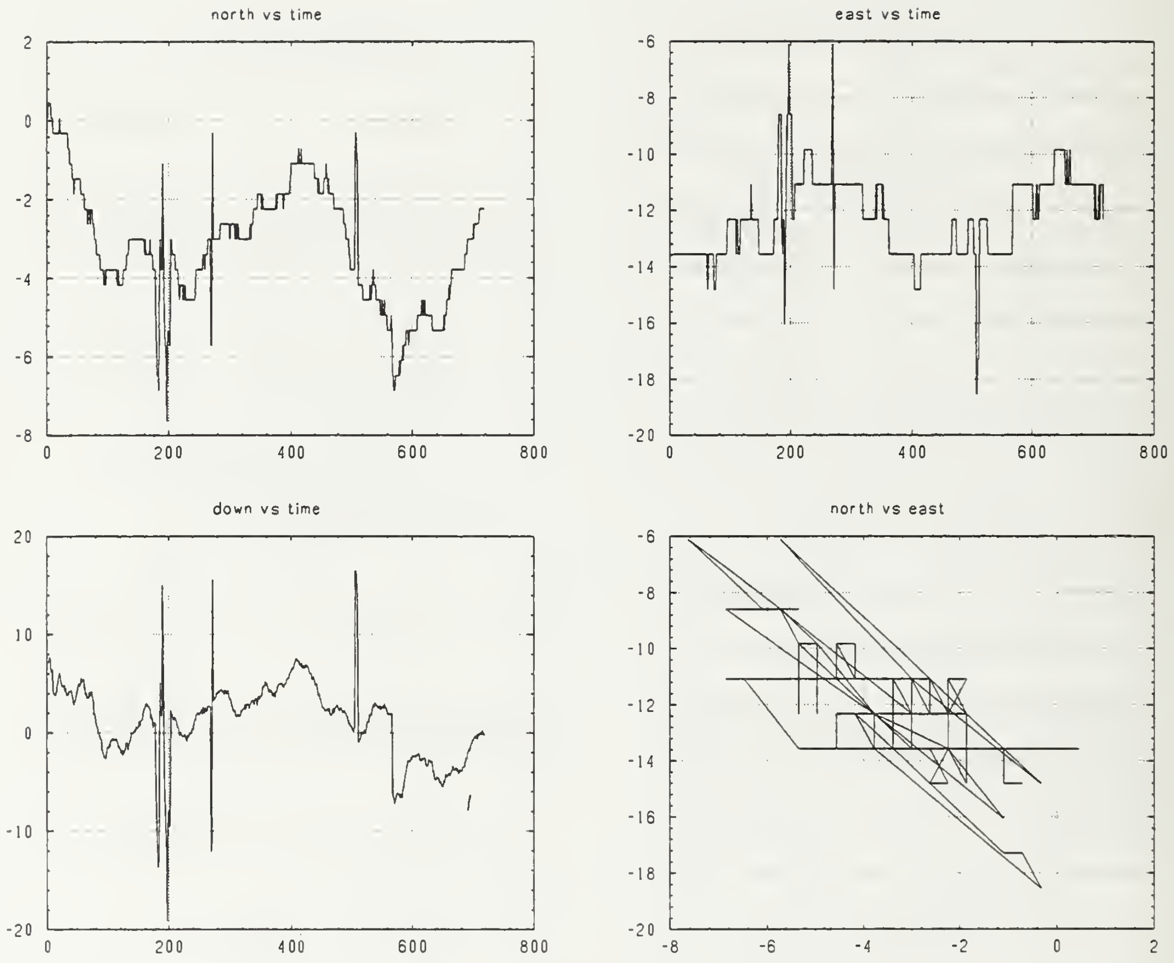

Figure 5.3: LTP data for run #3, all vertical axes in meters

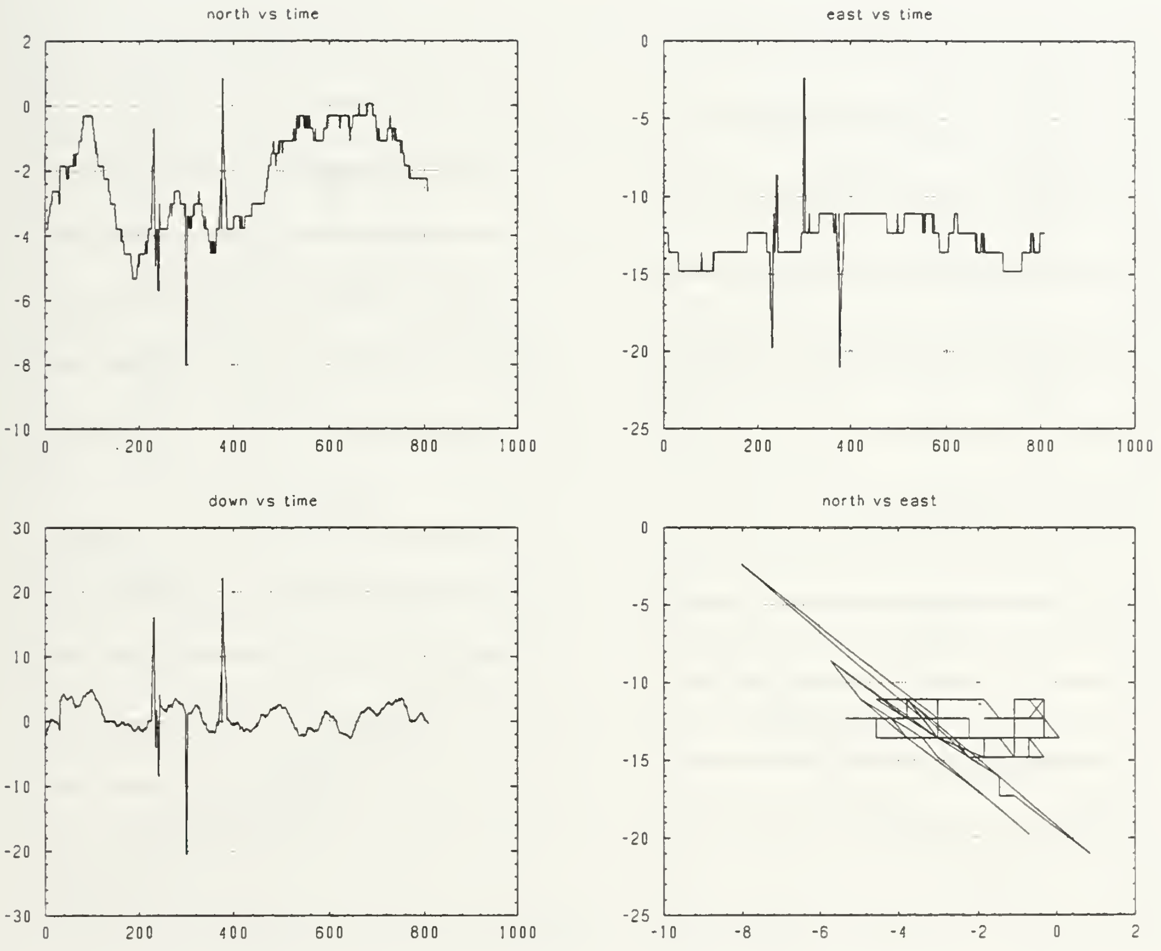

Figure 5.4: LTP data for run #4, all vertical axes in meters

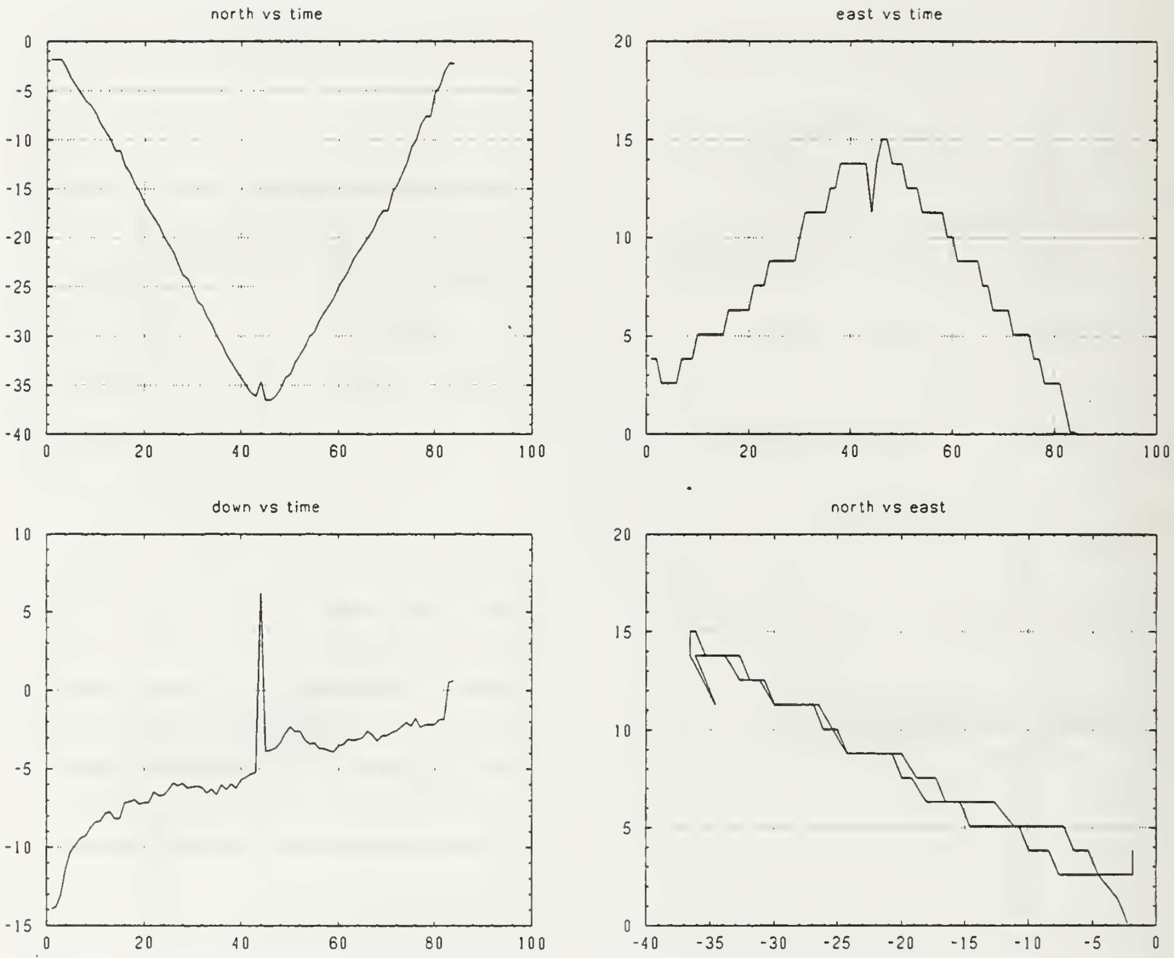

Figure 5.5: LTP data for run 5a, all vertical axes in meters

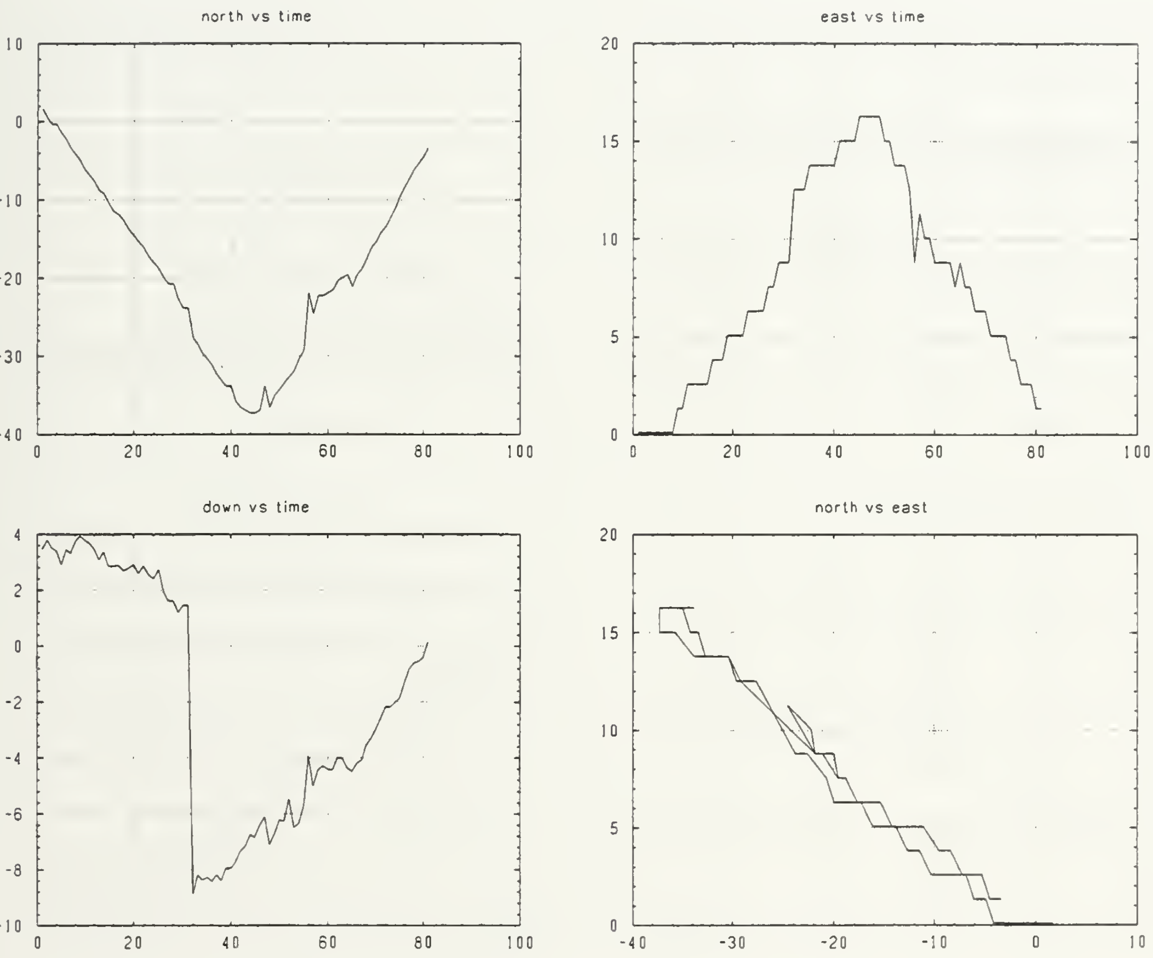

Figure 5.6: LTP data for run #6a, all vertical axes in meters

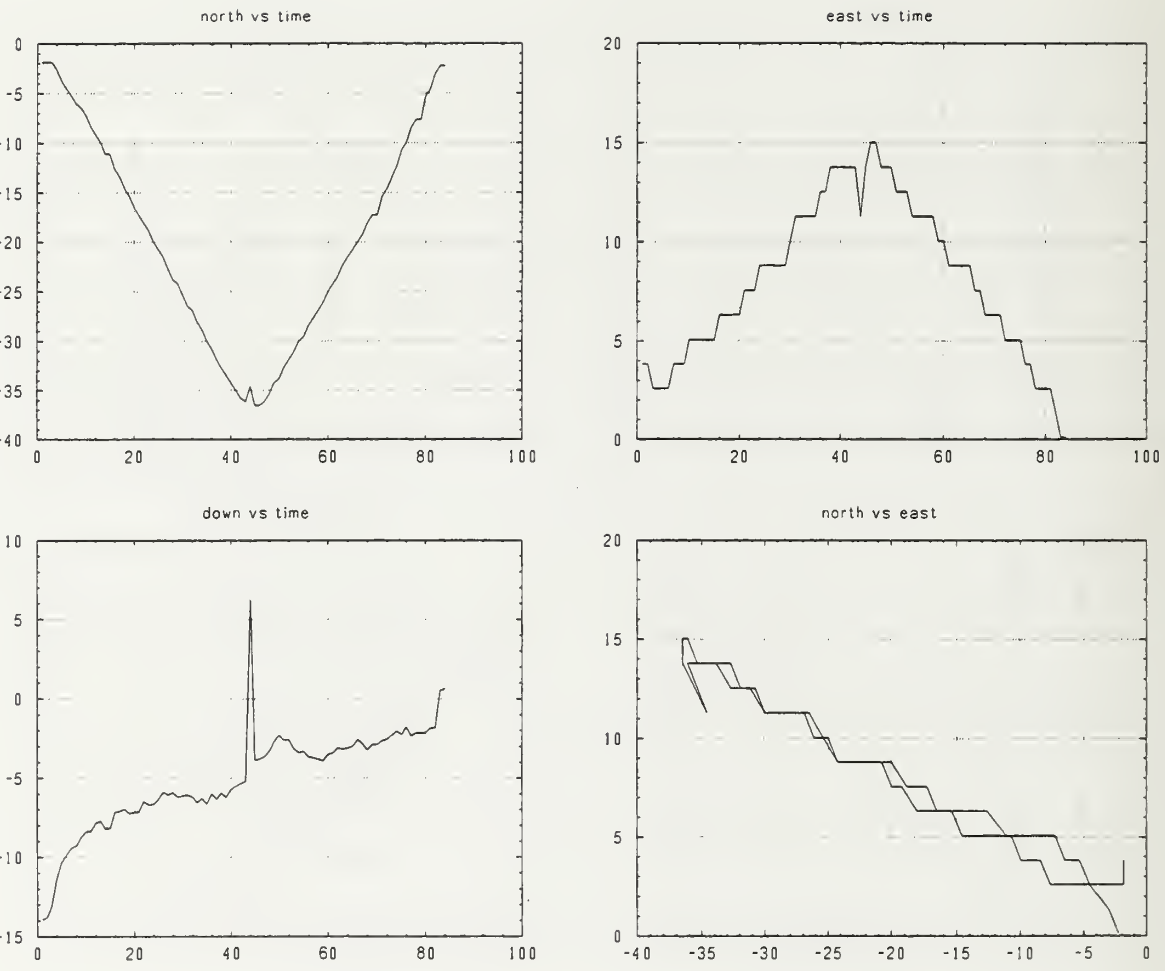

Figure 5.7: LTP data for run #5b, all vertical axes in meters

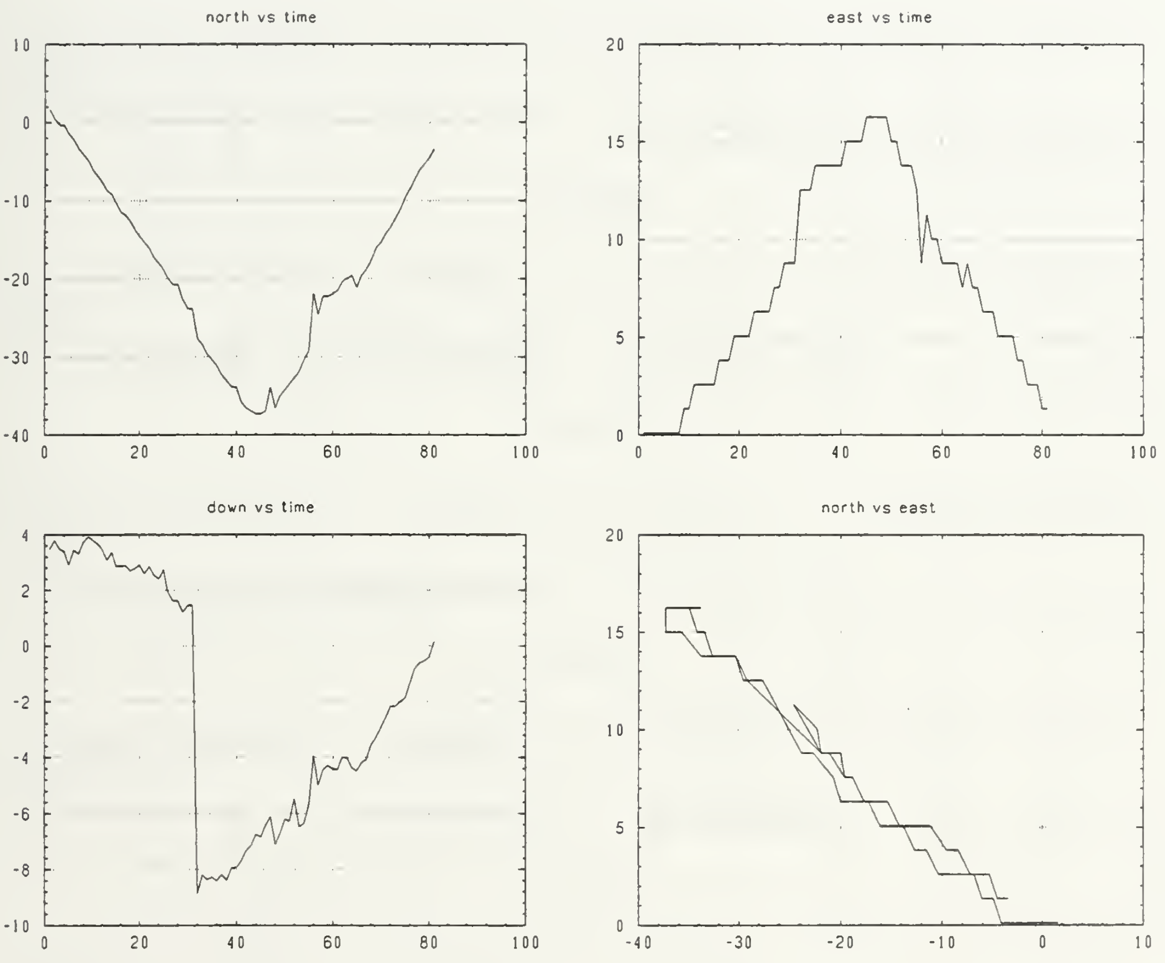

Figure 5.8: LTP data for run #6b, all vertical axes in meters

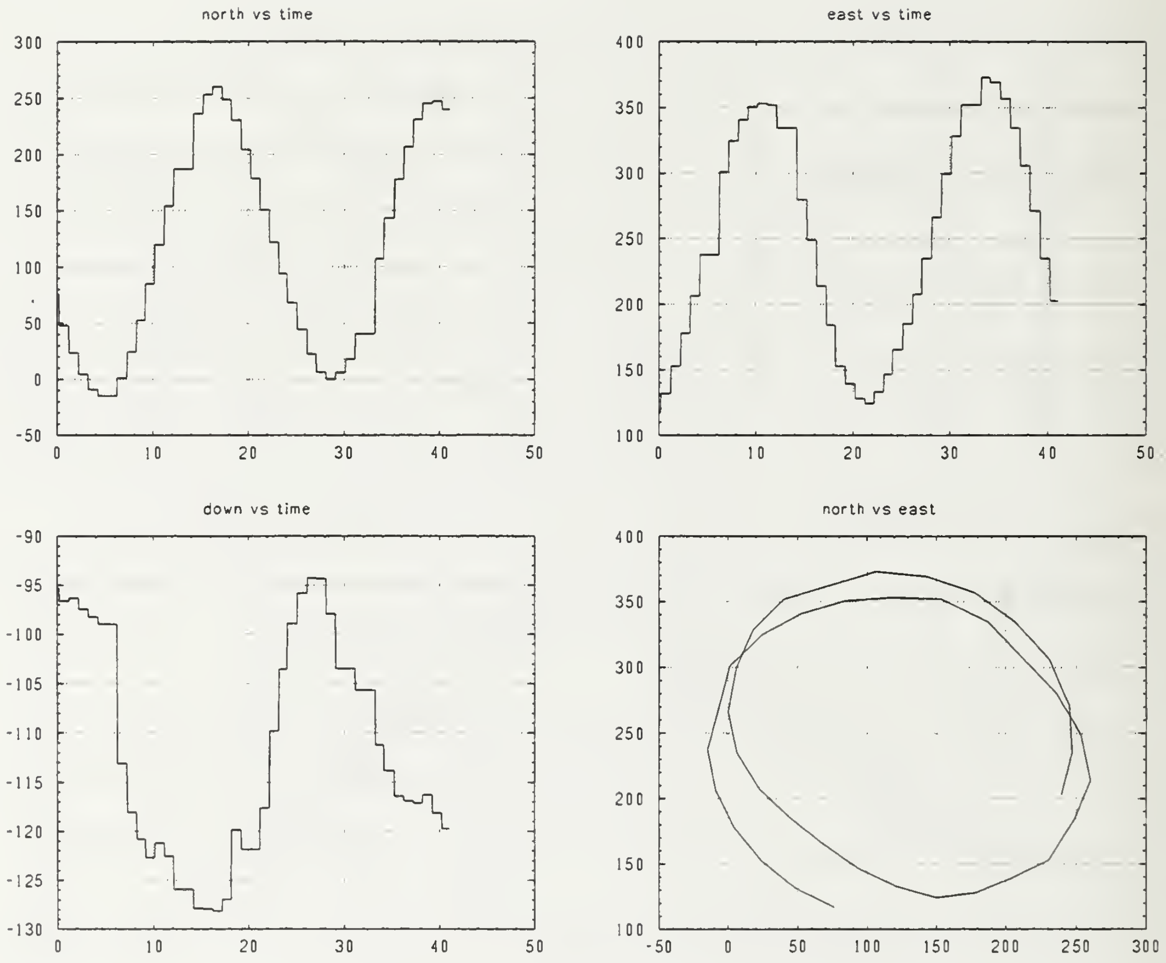

Figure 6.1: LTP graphs for flight test on 31 January, all distances in meters

APPENDIX B. PROFESSOR CLYNCH'S DATA

Here is the list of coordinates obtained from Professor Clynch in the Oceanography Department. These points are all in the parking lot of the Mechanical Engineering building.

NPS-5001

Latitude: 36.594263825° Longitude: -121.877218747° Altitude: -19.368m
{ECEF}
X: -2707.544794m Y: -4353.710077m Z: 3781.326527m

NPS-5002

Latitude: 36.594318186° Longitude: -121.876852525° Altitude: -19.058m
{ECEF}
X: -2707.515098m Y: -4353.724540m Z: 3781.331555m

NPS-5003

Latitude: 36.594866908° Longitude: -121.877366719° Altitude: -20.088
{ECEF}
X: -2707.534563m Y: -4353.668714m Z: 3781.379829m

NPS-5004

Latitude: 36.594590992° Longitude: -121.877094053° Altitude: -19.458
{ECEF}
X: -2707.523751m Y: -4353.697529m Z: 3781.355622m

NPS-5005

Latitude: 36.594888269° Longitude: -121.877237275° Altitude: -19.938
{ECEF}
X: -2707.524045m Y: -4353.673733m Z: 3781.381822m

LTP DATA FOR VARIOUS COMBINATIONS OF POINTS FROM PROF. CLYNCH

(All distances are in meters, LTP is defined as {ENU})

Origin: 5001	Data point: 5002	
E: 32.770	N: 6.032	U: 0.310
Origin: 5001	Data point: 5003	
E: -13.241	N: 66.924	U: -0.720
Origin: 5002	Data point: 5004	
E: -21.612	N: 30.273	U: -0.400
Origin: 5003	Data point: 5005	
E: 11.583	N: 2.370	U: 0.150
Origin: 5004	Data point: 5005	
E: -12.816	N: 32.989	U: -0.480
Origin: 5005	Data point: 5002	
E: 34.428	N: -63.262	U: 0.880

APPENDIX C. DATA VERIFICATION CODE

The following XMATH code takes the surveyed latitude and longitude of two points in the Mechanical Engineering parking lot of the Naval Postgraduate School, and converts this to ECEF and LTP coordinates for verification of the equations to be used in the RealSim controller.

```
#                                   -- clynchck.ms --
#        -- This program is to be used to confirm the below
#        -- equations by comparison with data from Prof. Cylnch.
#        -- Using data points in  ME parking lot from Prof. Clynch

#        -- Here is the coordinate transformation matrix from ECEF to
#        -- {t} = {UEN}

# C = [ cos(phi)*cos(lambda)   cos(phi)*sin(lambda)   sin(phi);
#        -sin(lambda)              cos(lambda)           0;
#        -sin(phi)*cos(lambda)  -sin(phi)*sin(lambda)  cos(phi)]

#        -- This is the xmath clear command
delete *.*

#                  -- Turn on the diary file called e2tang.dat --
# set sessiondiary "e2tang.dat"

#        -- This next section is for entering the origin in lat and lon
#        -- Altitude needs to be entered in meters in Geoid (Ellipsoidal)
#        -- height.

# p0_lat = 36.594263825?
# p0_lon = -121.877218747?
# h = 13.51

#        -- This time use pt NPS 5001 as the origin.
# p0_lat = 36.594263825
# p0_lon = -121.877218747
# h = -19.368

#        -- Here is the data for pt NPS 5004
 p0_lat = 36.594590992
 p0_lon = -121.877094053
 h = -19.458
```

```
#       -- Here is the data for pt NPS 5005 to be used as the origin.
# p0_lat = 36.594888269
# p0_lon = -121.877237275
# h = -19.938

#       -- To get origin in radians, multiply degrees by (pi/180)
p0_lon_r = p0_lon*0.01745
p0_lat_r = p0_lat*0.01745

#       -- Here the lat and lon of the origin will be converted
#       -- to ECEF coordinates {x,y,z}.
s_phi = sin(p0_lat_r)
c_phi = cos(p0_lat_r)
s_lambda = sin(p0_lon_r)
c_lambda = cos(p0_lon_r)

#       -- 'N' and 'Eps_sq' are conversion factors for a non-spherical
#       -- Earth. 'N' is in meters.
#       -- `a` is the semimajor axis of the Earth in meters
Eps_sq = 0.00669437999013
a = 6378137
den = (1-(Eps_sq*(s_phi)^2))^0.5
N = a/den

p0_x = (N+h)*c_phi*c_lambda
p0_y = (N+h)*c_phi*s_lambda
p0_z = ((N*(1-Eps_sq))+h)*s_phi

#       -- This section is for calculating position in LTP.
#       -- Here is where the aircraft position will be fed in by the
#       -- receiver as lat and lon and alt and we will convert it to
#       -- ECEF {xyz}
# pt2_lat = 36.55020693
# pt2_lon = -121.54329221
# h2 = -5.67

#       -- Here is the data for pt NPS 5002
 pt2_lat = 36.594318186
 pt2_lon = -121.876852525
 h2 = -19.058

#       -- Here is the data for pt NPS 5003
# pt2_lat = 36.594866908
# pt2_lon = -121.877366719
# h2 = -20.088

#       -- Here is the data for pt NPS 5004
# pt2_lat = 36.594590992
# pt2_lon = -121.877094053
# h2 = -19.458

#       -- Here is the data for pt NPS 5005
# pt2_lat = 36.594888269
# pt2_lon = -121.877237275
# h2 = -19.938

#       -- Now make necessary computations for pt2.
        pt2_lon_r = pt2_lon*0.01745
        pt2_lat_r = pt2_lat*0.01745

        s_phi2 = sin(pt2_lat_r)
        c_phi2 = cos(pt2_lat_r)
        s_lambda2 = sin(pt2_lon_r)
        c_lambda2 = cos(pt2_lon_r)

        pt2_x = (N+h2)*c_phi2*c_lambda2
        pt2_y = (N+h2)*c_phi2*s_lambda2
        pt2_z = ((N*(1-Eps_sq))+h2)*s_phi2

#       -- Now to get the difference between the two points
#       -- in ECEF {xyz}.
        del_x = pt2_x - p0_x
        del_y = pt2_y - p0_y
        del_z = pt2_z - p0_z

        del = [del_x; del_y; del_z]

#       -- The angle Lambda is the same as the longitude, while the angle
#       -- Phi is the same as the latitude.
        lambda = p0_lon_r
        phi = p0_lat_r

#       -- Now we want to convert to a {NED} coordinate system.
        C = [ -sin(phi)*cos(lambda),  -sin(phi)*sin(lambda),  cos(phi);
          -sin(lambda),                cos(lambda),           0;
          -cos(phi)*cos(lambda),  -cos(phi)*sin(lambda),  -sin(phi)]
```

```
#        -- The rotation matrix (C) times the delta {x,y,z} will give
#        -- the aircraft's current position in the tangent plane in
#        -- meters.
         NED = [C*del]'?
```

APPENDIX D. DATA TRANSFORMATION VERIFICATION

Below is the result of running Prof. Clynch's ME parking lot coordinates through the program "Clynchck.ms". This program has the same coordinate transformation routine in it as is in the RealSim block diagram program implemented into the UAV controller. This was done in order to compare the resulting values to Prof. Clynch's values in Appendix C, and thus validate the algorithm. This was successfully accomplished.

Each section will start with an origin point and a distant point to be converted to ECEF {x,y,z} and also to LTP {NED}. All latitude and longitude values are in degrees, while all ECEF and LTP values are in meters.

-- The origin is pt 5004, and pt2 is 5002 --

```
p0_lat =  36.59459099200000

p0_lon =   -1.218770940530000e+002

h = -19.45800000000000

p0_x =   -2.707523751060266e+006

p0_y =   -4.353697528851314e+006

p0_z =    3.781355622141596e+006

pt2_lat =  36.59431818600000

pt2_lon =   -1.218768525250000e+002

h2 = -19.05800000000000

pt2_x =   -2.707515139893124e+006

pt2_y =   -4.353724607049335e+006

pt2_z =    3.781331612821002e+006

NED = -30.27349302602244  21.61242196167804  -0.49731272875636
```

-- Here the origin is pt 5005 and pt2 is 5002 --

```
p0_lat =  36.59488826900000

p0_lon =   -1.218772372750000e+002

h = -19.93800000000000

p0_x =   -2.707524044669156e+006

p0_y =   -4.353673733355007e+006

p0_z =    3.781381821735875e+006

pt2_lat =  36.59431818600000

pt2_lon =   -1.218768525250000e+002

h2 = -19.05800000000000

pt2_x =   -2.707515185012028e+006

pt2_y =   -4.353724679601202e+006

pt2_z =    3.781331675834331e+006

NED = -63.26254233972134  34.42822152785000  -1.08317434025619

          -- p0 is pt 5001 and pt2 is 5003 --

p0_lat =  36.59426382500000

p0_lon =   -1.218772187470000e+002

h = -19.36800000000000

p0_x =   -2.707544694196697e+006

p0_y =   -4.353710077013256e+006

p0_z =    3.781326526895528e+006

pt2_lat =  36.59486690800000

pt2_lon =   -1.218773667190000e+002
```

```
h2 = -20.08800000000000

pt2_x =   -2.707534471734960e+006

pt2_y =   -4.353668566997226e+006

pt2_z =    3.781379701332720e+006

NED =  66.92464475696794 -13.24074221985020   0.93573232907061

          -- Here p0 is 5002 and pt2 is 5001

p0_lat =  36.59431818600000

p0_lon =   -1.218768525250000e+002

h = -19.05800000000000

p0_x =   -2.707515098488406e+006

p0_y =   -4.353724540469932e+006

p0_z =    3.781331554994929e+006

pt2_lat =  36.59426382500000

pt2_lon =   -1.218772187470000e+002

h2 = -19.36800000000000

pt2_x =   -2.707544702447331e+006

pt2_y =   -4.353710090280208e+006

pt2_z =    3.781326538418272e+006

NED =  -6.03242469894091 -32.77031841760502   0.29067419837140

          -- p0 is 5005 and pt2 is 5003

p0_lat =  36.59488826900000

p0_lon =   -1.218772372750000e+002
```

```
h = -19.93800000000000

p0_x =   -2.707524044669156e+006

p0_y =   -4.353673733355007e+006

p0_z =    3.781381821735875e+006

pt2_lat =  36.59486690800000

pt2_lon =   -1.218773667190000e+002

h2 = -20.08800000000000

pt2_x =   -2.707534566509818e+006

pt2_y =   -4.353668719393558e+006

pt2_z =    3.781379833696587e+006

NED =  -2.37044078542388 -11.58283118250856   0.14238271532010

           -- p0 is 5005 and pt2 is 5004

p0_lat =  36.59488826900000

p0_lon =   -1.218772372750000e+002

h = -19.93800000000000

p0_x =   -2.707524044669156e+006

p0_y =   -4.353673733355007e+006

p0_z =    3.781381821735875e+006

pt2_lat =  36.59459099200000

pt2_lon =   -1.218770940530000e+002

h2 = -19.45800000000000

pt2_x =   -2.707523796179318e+006
```

```
pt2_y =   -4.353697601402736e+006

pt2_z =    3.781355685155328e+006

NED = -32.98907843753755  12.81575309481910  -0.58606205866278
```

APPENDIX E. COORDINATE TRANSFORMATION CODE

This appendix contains the XMATH code used to verify the coordinate transformations and then implemented into the RealSim block diagram environment. This code will take output data from the *AC100* flight controller and transform the latitude, longitude, and geoid height coordinates to a Local Tangent Plane with a north, east, down orientation.

```
#                  -- e2tang.ms --

#         -- This function converts a vector given in the ECEF coordinate
#         -- system into the tangent plane
#
#         -- Using data points in  ME parking lot from Prof. Clynch

#         -- This is the xmath clear command
#  delete *.*
set format="long"

#                  -- Turn on the diary file called e2tang.dat --
# set sessiondiary "run8b.dat"

#         -- This next section is for entering the origin in lat and lon
#         -- Altitude needs to be entered in meters in Geoid (Ellipsoidal)
#         -- height.

#         -- Here is the data for pt NPS 5005 to be used as the origin.

p0_lat = 36.594888269
p0_lon = -121.877237275
h = -19.938

#         -- To get origin in radians, multiply degrees by (pi/180)

p0_lon_r = p0_lon*(pi/180)
p0_lat_r = p0_lat*(pi/180)

#         -- Here the lat and lon of the origin will be converted
#         -- to ECEF coordinates {x,y,z}.

s_phi = sin(p0_lat_r)
c_phi = cos(p0_lat_r)
s_lambda = sin(p0_lon_r)
```

```
c_lambda = cos(p0_lon_r)

#       -- 'N' and 'Eps_sq' are conversion factors for a non-spherical
#       -- Earth. 'N' is in meters.
#       -- `a` is the semimajor axis of the Earth in meters

Eps_sq = 0.00669437999013
a = 6378137
den = (1-(Eps_sq*(s_phi)^2))^0.5
N = a/den

p0_x = (N+h)*c_phi*c_lambda?
p0_y = (N+h)*c_phi*s_lambda?
p0_z = ((N*(1-Eps_sq))+h)*s_phi?

#       -- This section is for calculating position in LTP.
#       -- Here is where the aircraft position will be fed in by the
#       -- receiver as lat and lon and alt and we will convert it to
#       -- ECEF {xyz}

#       -- Use this section to feed in a string of stored data points

# load "flight_test_rf_2.dat"
# load "flight_test_rf_3.dat"
# load "flight_test_rf_4.dat"
# load "flight_test_rf_5.dat"
# load "flight_test_rf_6.dat"
# load "flight_test_rf_7.dat"
# load "flight_test_rf_8.dat"
# load "flight_test_rf_9.dat"
# load "flight_test_rf_10.dat"
# load "flight_test_rf_14_m1.dat"

#       -- Define the matrices which will be used to plot data from

NED_meters = []
pt2_lat_vec = []
pt2_lon_vec = []
h2_vec = []
pt2_x_vec = []
pt2_y_vec = []
pt2_z_vec = []
```

```
#       -- Set up loop to read in data and put into matrices
i = 1
while i <= length(lat)

#       -- convert deciseconds to degrees

        pt2_lat = lat(i)/36000
        pt2_lon = lon(i)/36000
        h2 = hei_gps(i)

#       -- Now make necessary computations for pt2.

        pt2_lon_r = pt2_lon*(pi/180)
        pt2_lat_r = pt2_lat*(pi/180)

        s_phi2 = sin(pt2_lat_r)
        c_phi2 = cos(pt2_lat_r)
        s_lambda2 = sin(pt2_lon_r)
        c_lambda2 = cos(pt2_lon_r)

        den = (1-(Eps_sq*(s_phi2)^2))^0.5
        N = a/den

        pt2_x = (N+h2)*c_phi2*c_lambda2
        pt2_y = (N+h2)*c_phi2*s_lambda2
        pt2_z = ((N*(1-Eps_sq))+h2)*s_phi2

        pt2_x_vec = [pt2_x_vec;pt2_x]
        pt2_y_vec = [pt2_y_vec;pt2_y]
        pt2_z_vec = [pt2_z_vec;pt2_z]

#       -- Now to get the difference between the two points
#       -- in ECEF {xyz}.

        del_x = pt2_x - p0_x
        del_y = pt2_y - p0_y
        del_z = pt2_z - p0_z

        del = [del_x; del_y; del_z]

#       -- The angle Lambda is the same as the longitude, while the angle
#       -- Phi is the same as the latitude.
```

```
        lambda = p0_lon_r
        phi = p0_lat_r




#       -- Now we want to convert to a {NED} coordinate system.

        C = [ -sin(phi)*cos(lambda),  -sin(phi)*sin(lambda),  cos(phi);
          -sin(lambda),                  cos(lambda),         0;
          -cos(phi)*cos(lambda),  -cos(phi)*sin(lambda),  -sin(phi)]

#       -- The rotation matrix (C) times the delta {x,y,z} will give
#       -- the aircraft's current position in the tangent plane in
#       -- meters.

        NED = [C*del]'
        NED_meters = [NED_meters;NED]
        pt2_lat_vec = [pt2_lat_vec;pt2_lat]
        pt2_lon_vec = [pt2_lon_vec;pt2_lon]
        h2_vec = [h2_vec;h2]

        i = i+1
endwhile

#       -- Now to split the data into three groups.

north = NED_meters(:,1)
east = NED_meters(:,2)
down = NED_meters(:,3)
#
#       -- The following sigma and mean are only if the data is for
#       -- a point, not a line.
#
#       -- First is lat and lon
lat_mean_pt = mean(pt2_lat_vec)?
lat_sigma_pt = sqrt(variance(pt2_lat_vec))?

lon_mean_pt = mean(pt2_lon_vec)?
lon_sigma_pt = sqrt(variance(pt2_lon_vec))?

alt_mean_pt = mean(h2_vec)?
alt_sigma_pt = sqrt(variance(h2_vec))?
```

```
#        -- Now for x, y, z

x_mean_pt = mean(pt2_x_vec)?
x_sigma_pt = sqrt(variance(pt2_x_vec))?

y_mean_pt = mean(pt2_y_vec)?
y_sigma_pt = sqrt(variance(pt2_y_vec))?

z_mean_pt = mean(pt2_z_vec)?
z_sigma_pt = sqrt(variance(pt2_z_vec))?

#        -- Now for NED

north_mn_pt = mean(north)?
north_sigma_pt = sqrt(variance(north))?

east_mn_pt = mean(east)?
east_sigma_pt = sqrt(variance(east))?

down_mn_pt = mean(down)?
down_sigma_pt = sqrt(variance(down))?

#        -- Here are the various plots

plot_ll = plot(pt2_lat_vec,{rows=2,columns=2,title="lat vs time"})
plot_ll = plot(pt2_lon_vec,{keep=plot_ll,graph_number=2,title="lon vs time"})
plot_ll = plot(h2_vec,{keep=plot_ll,graph_number=3, title="alt vs time"})
plot_ll = plot(pt2_lat_vec,pt2_lon_vec,{keep=plot_ll,graph_number=4,title="lat vs lon"})?

plot_ned = plot(north,{rows=2,columns=2,title="north vs time"})
plot_ned = plot(east,{keep=plot_ned,graph_number=2,title="east vs time"})
plot_ned = plot(down,{keep=plot_ned,graph_number=3,title="down vs time"})
plot_ned = plot(north,east,{keep=plot_ned,graph_number=4,title="north vs east"})?

plot_ne_ll = plot(north,east,{rows=1,columns=2,title="north vs east"})
plot_ne_ll = plot(pt2_lat_vec,pt2_lon_vec,{keep=plot_ne_ll,graph_number=2,title="lat vs
lon"})?

plot_xyz = plot(pt2_x_vec,{rows=2,columns=2,title="x vs time"})
plot_xyz = plot(pt2_y_vec,{keep=plot_xyz,graph_number=2,title="y vs time"})
plot_xyz = plot(pt2_z_vec,{keep=plot_xyz,graph_number=3,title="z vs time"})
plot_xyz = plot(pt2_x_vec,pt2_y_vec,{keep=plot_xyz,graph_number=4,title="x vs y"})?
```

```
plot_3d = plot(north,east,down,{rows=1,columns=2,title="north vs east vs down"})
plot_3d = plot(pt2_lat_vec,pt2_lon_vec,h2_vec,{keep=plot_3d,graph_number=2,title="lat vs lon
vs h"})?

#
#       -- Need to determine the mean and std deviation
#       -- from a line between the 2 known inputs
#       -- This is for lat and lon
#

#       -- First divide straight line into same number of points
#       -- as data has
#       -- y is number of pts only need line going one way, not both.
#       -- v is starting data pt number, w is ending data pt number
 v = 45
#  w = 45
 w = length(h2_vec)
 z = (v:w)
 y = length(z)
 x = ones(y,1)

ptA_x = pt2_x_vec(v)
ptA_y = pt2_y_vec(v)
ptA_z = pt2_z_vec(v)

ptB_x = pt2_x_vec(w)
ptB_y = pt2_y_vec(w)
ptB_z = pt2_z_vec(w)

del_x = (ptB_x - ptA_x)/(y-1)
del_y = (ptB_y - ptA_y)/(y-1)
del_z = (ptB_z - ptA_z)/(y-1)

x_line = []
y_line = []
z_line = []

#
#       -- Now create evenly spaced points on the line
#
for j = 1:y
        x_pt=ptA_x+del_x*(j-1)
        x_line = [x_line;x_pt]
        y_pt=ptA_y+del_y*(j-1)
        y_line = [y_line;y_pt]

        z_pt=ptA_z+del_z*(j-1)
        z_line = [z_line;z_pt]
endfor

x_diff = pt2_x_vec(v:w) - x_line
y_diff = pt2_y_vec(v:w) - y_line
z_diff = pt2_z_vec(v:w) - z_line

#       -- The following sigma and mean are only valid if the data
#       -- is for a line and not for a point.
#
x_mean_path = mean(x_diff)?
y_mean_path = mean(y_diff)?
z_mean_path = mean(z_diff)?

x_sigma_path = sqrt(variance(x_diff))?
y_sigma_path = sqrt(variance(y_diff))?
z_sigma_path = sqrt(variance(z_diff))?
#
#       -- Now for lat and lon
#
ptA_lat = pt2_lat_vec(v)
ptA_lon = pt2_lon_vec(v)
ptA_alt = h2_vec(v)

ptB_lat = pt2_lat_vec(w)
ptB_lon = pt2_lon_vec(w)
ptB_alt = h2_vec(w)

del_lat = (ptB_lat - ptA_lat)/(y-1)
del_lon = (ptB_lon - ptA_lon)/(y-1)
del_alt = (ptB_alt - ptA_alt)/(y-1)

lat_line = []
lon_line = []
alt_line = []
#
#       -- Now create evenly spaced points on the line
#
```

```
for j = 1:y
        lat_pt=ptA_lat+del_lat*(j-1)
        lat_line = [lat_line;lat_pt]

        lon_pt=ptA_lon+del_lon*(j-1)
        lon_line = [lon_line;lon_pt]

        alt_pt=ptA_alt+del_alt*(j-1)
        alt_line = [alt_line;alt_pt]
endfor

lat_diff = ((lat(v:w))/36000) - lat_line
lon_diff = ((lon(v:w))/36000) - lon_line
alt_diff = h2_vec(v:w) - alt_line

lat_mean_path = mean(lat_diff)?
lon_mean_path = mean(lon_diff)?
alt_mean_path = mean(alt_diff)?

lat_sigma_path = sqrt(variance(lat_diff))?
lon_sigma_path = sqrt(variance(lon_diff))?
alt_sigma_path = sqrt(variance(alt_diff))?

#       -- Now to develop sigma and mean for LTP
#       -- Since line is out and back, we only need to look at
#       -- one direction at a time. So 'y' will be turnaround
#       -- pt.

pt2A_n = north(v)
pt2B_n = north(w)

pt2A_e = east(v)
pt2B_e = east(w)

pt2A_d = down(v)
pt2B_d = down(w)

dist_ab_n = pt2B_n - pt2A_n
dist_ab_e = pt2B_e - pt2A_e
dist_ab_d = pt2B_d - pt2A_d

deln_ab = dist_ab_n/(y-1)
dele_ab = dist_ab_e/(y-1)
```

```
deld_ab = dist_ab_d/(y-1)

n_line = []
e_line = []
d_line = []

for k = 1:y
        n_pt = pt2A_n + deln_ab*(k-1)
        e_pt = pt2A_e + dele_ab*(k-1)
        d_pt = pt2A_d + deld_ab*(k-1)

        n_line = [n_line;n_pt]
        e_line = [e_line;e_pt]
        d_line = [d_line;d_pt]
endfor


#       -- create vectors of the original data of the same
#       -- length as the line I am trying to match

n_vec = north([v:1:w])
e_vec = east([v:1:w])
d_vec = down([v:1:w])

n_diff = n_vec - n_line
e_diff = e_vec - e_line
d_diff = d_vec - d_line
#
#       -- Mean error and sigma for each of 3 directions in LTP
#
n_mean_path = mean(n_diff)?
e_mean_path = mean(e_diff)?
d_mean_path = mean(d_diff)?

n_sigma_path = sqrt(variance(n_diff))?
e_sigma_path = sqrt(variance(e_diff))?
d_sigma_path = sqrt(variance(d_diff))?

plot_vec = plot(n_vec,{rows=2,columns=2,title="north vs time"})
plot_vec = plot(n_vec,e_vec,{keep=plot_vec,graph_number=2,title="north vs east"})
plot_vec = plot(n_line,{keep=plot_vec,graph_number=3,title="line from A to B"})?

plot_over = plot(n_line,e_line,{title="line vs data"})
plot_over = plot(n_vec,e_vec,{keep=plot_over})?

mag_vec = ((n_diff)^(2) + (e_diff)^(2) + (d_diff)^(2))^(.5)
dist_ab_n = pt2B_n - pt2A_n
dist_ab_e = pt2B_e - pt2A_e
dist_ab_d = pt2B_d - pt2A_d

unit_vec = [(dist_ab_n/mag_vec),(dist_ab_e/mag_vec),(dist_ab_d/mag_vec)]'
diff_vec = [n_diff,e_diff,d_diff]

#       -- Need to sum diffs and take norm in order to help in
#       -- developing the 3-D sigma and mean which also require
#       -- the unit vector of the straight line.

line_mag = ((dist_ab_n)^(2) + (dist_ab_e)^(2) + (dist_ab_d)^(2))^(.5)
unit_vec_n = dist_ab_n/line_mag
unit_vec_e = dist_ab_e/line_mag
unit_vec_d = dist_ab_d/line_mag

unit_cross_pt = []
for k = 1:y

        pt_vec = [n_diff(k); e_diff(k); d_diff(k)]
        unit_vec = [unit_vec_n, unit_vec_e, unit_vec_d]

#
#       -- To compute the cross product, use the skew symmetric matrix
#       -- for 3-d_dist_from_line = unit_vec CROSS pt_vec
#

        t = [0,         -unit_vec_d, unit_vec_e;
            unit_vec_d,  0,          -unit_vec_n;
            -unit_vec_e, unit_vec_n,  0]
        cross_prod = [t * (pt_vec)]'
        unit_cross_pt = [unit_cross_pt;cross_prod]
        k=k+1
endfor
unit_cross_pt

#       -- Set up to take the norm of each of the pts(all 3 directions)
#       -- in order to obtain a 3d mag of the error

#       -- mag_dist_vec will be a vector with the 3-d magnitude of the distance
```

```
#	-- from the data pts to the straight line, this allows for
#	-- non-constant velocity.

mag_dist_vec = []
for k = 1:y
	mag_dist = ((unit_cross_pt(k,1))^(2) + (unit_cross_pt(k,2))^(2) +
(unit_cross_pt(k,3))^(2))^(0.5)
	mag_dist_vec = [mag_dist_vec; mag_dist]
	k = k + 1
endfor

mag_dist_vec?
mean_3d = mean(mag_dist_vec)?

sigma_3d = sqrt(variance(mag_dist_vec))?

#	-- Don't forget to turn off the diary file
# remove sessiondiary
```

LIST OF REFERENCES

1. Dana, Peter H., The Geographer's Craft Project, Department of Geography, The University of Texas at Austin, [web page] http://wwwhost.cc.utexas.edu/ftp/pub/grg/gcraft/notes/gps/gps.html#contents

2. Zanino, J. A., *Uniform system for the Rapid Prototyping and Testing of Controllers for Unmanned Aerial Vehicles,* Master's Thesis, Naval Postgraduate School, Monterey, CA, 1996.

3. Kaminer, I., *AA3276: Intro to Avionics,* Course Notes, Naval Postgraduate School, Monterey, CA, Sep 1996.

4. Clynch, J. R., *GPS Accuracy Levels,* Notes, Naval Postgraduate School, Monterey, CA, May 1996.

5. Motorola, *Motorola GPS Quick Start Guide,* Revision 3.0, March 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Doctor Isaac I. Kaminer, Code AA/KA 3
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, California 93943-5121

4. Doctor Richard M. Howard, Code AA/HO 1
   Department of Aeronautics and Astronautics
   Naval Postgraduate School
   Monterey, California 93943-5121

5. Department of Aeronautics and Astronautics 1
   Code AA
   Naval Postgraduate School
   699 Dyer Rd. Rm. 137
   Monterey, California 93943-5106

6. Lieutenant Peyton M. Allen 3
   2215 Bear Creek Rd.
   Kerrville, Texas 78028